Elektronica en Telecommunicatie 10

Gemeenschappelijke publikatie van de
Sectie voor Telecommunicatietechniek van het K.I.v.I. en het
Nederlands Elektronica- en Radiogenootschap.
Redactie-adres: Prinsessegracht 23, Den Haag.

681.325.6:621.382.049.7

Programmed Accumulator

by prof. dr. ir. R. M. M. Oberman, Switching Laboratory, Department of Electrical Engineering, T.H. Delft



Synopsis: Apart from the always present logical operations, the organization of a digital circuit will, for an important part, be based on the operations 'shift', 'count', and 'add'. The first two operations are sequential, the last one is combinational. In this paper the problem is discussed of programming a general type of circuit such that it can perform these operations.

The still rapidly increasing number of different types of integrated circuits (I.C.'s) in the field of shift registers, counters, adders, etc., creates a wide field of application for a general-purpose type of integrated circuit that can replace the several specialized types. It will be shown, that the *accumulator*, which in fact is a combination of a 'full adder' and a 'master-slave type memory element', can be used as such a general-purpose type of I.C.

1. Introduction

In integrated circuits (I.C.'s) simple logical functions are implemented by means of rather complicated circuits, compared with circuits built from lumped components. Many types of I.C.'s, designed for one specific purpose only, are now commercially available; a few are designed for more than one function. For simple logical functions the problem of applying standard-type I.C.'s has already been treated by the author in another paper [1]. In the present paper the problem will be discussed of using a general-purpose circuit for designing shift registers, counters and adders. From the switching point of view, this type of circuit is far more complex than the quadruple standard gate discussed in [1].

It will be shown that the computer accumulator which stores a number and, upon reception of another number, adds the two numbers and then stores the sum, etc., can be operated as a multi-purpose circuit. In integrated form, this multi-purpose circuit can replace many of the already existing integrated circuits. However, it will be more complex than most of the existing integrated circuits designed to perform one function only. This disadvantage in economy must be compensated for by its much greater usefulness and hence by the much greater numbers in which it will be required and produced.

The increase in scale of integration during recent years has opened the possibility of TTL multi-purpose I.C.'s this paper is dealing with. A first step in this direction is the binary, programmable, cascadable, divide-by-*n* counter described in [3]. Another publication in this field [4] was issued at the early stage of preparation of this paper.

The latest steps are perhaps the announcement of production of a 4-bit TTL accumulator by INTEL and an 8-bit MOS accumulator by FAIRCHILD.

The background of our investigation was the fact that with printed circuits and lumped components a vast range of digital circuits could be designed by using only a few different standard cards. The standard cards contain the necessary basic logical and memory functions. In I.C.-design these basic functions can of course be obtained, but many more I.C.'s are now commercially available that are functionally orientated and it is not practical to leave these unused. As a result of this it will no longer be possible to continue designing digital circuits consisting of a few printed cards only. This has two disagreeable effects on the economy of digital circuits. Firstly, a substantial number of printed cards must be kept in store and secondly, a high price will have to be paid for those special I.C.'s, that have to be purchased in small numbers.

For not too complicated circuits a great number of different digital integrated circuits can be substituted by the standard gate without serious loss of economy in hardware [1]. In the following it will be shown that the four-bit accumulator can be used successfully as a general-purpose device in a class of more complicated digital circuits.

2. The Accumulator

An accumulator is a digital circuit that adds a number, fed to its inputs, to the number already stored in its memory [2, 5].* It is a circuit which contains full adder circuits and memory elements. A block diagram of a four-section accumulator is shown in Fig. 1. Each section of this block diagram contains

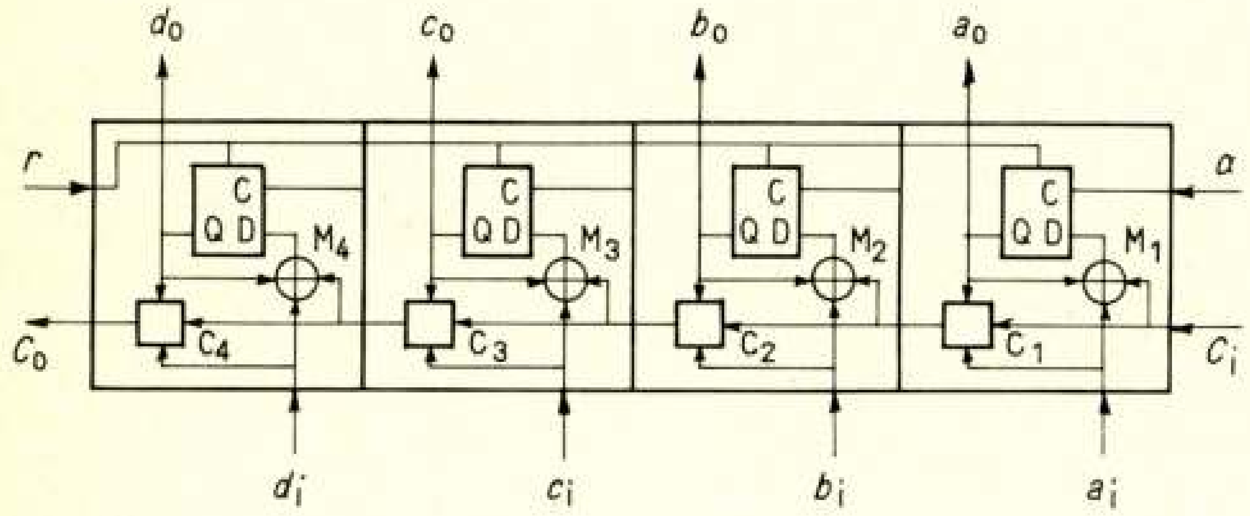


Fig. 1. Block diagram of a 4-bit accumulator.

a full adder, consisting of mod 2 adder M and carry-forming network C, and a D-type MS flip-flop as memory element. These circuits co-operate in such a way that number $d_i c_i b_i a_i$ fed to its inputs is added to number $d_o c_o b_o a_o$ already stored in the memory elements of the circuit. This adding operation will be repeated at each clock pulse α. For numbers with more than four digits, the accumulator can be cascaded with other identical circuit blocks. The carry output C_o of one block then has to be connected with input C_i of the next block.

Input terminal r represents a general reset which, in many types of memory circuits, operates independently of clock pulse α.

In the following sections the accumulator circuit of Fig. 1 will be represented by the simpler symbol of Fig. 2.

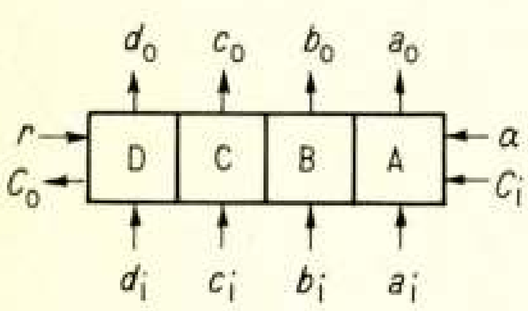


Fig. 2. 4-bit accumulator.

* In modern literature an accumulator is often defined to be a memory location, where the result of an operation is stored. However, the author's definition is based on the orginal interpretation of the accumulator concept.

2.1. *Binary Counter*

The accumulator of Fig. 2 can be made to operate as a synchronous natural binary counter by connecting input C_i to 0 and priming inputs $d_i c_i b_i a_i$ with 0001. The pulse series to be counted has to be fed to the α input.

The accumulator operates as follows. At each counting pulse $\alpha = 1$, a '1' is added to the content of the accumulator. The number of counting pulses α thus is automatically registered in the memory elements of the accumulator as a binary number.

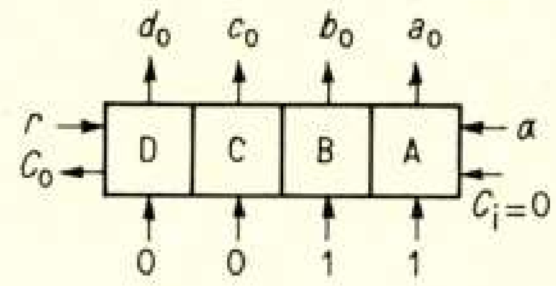


Fig. 3. Binary up-counter.

With $d_i c_i b_i a_i = 0001$ as priming number, the accumulator is an up-counter (Fig. 3). However, when the 'two's complement' of $d_i c_i b_i a_i = 0001$ is used as priming number, i.e. $d_i c_i b_i a_i = 1111 = -1$, the accumulator will operate as a down-counter. This operation is in fact a subtraction of 1 or an addition of -1 at each counting pulse α. The priming of the accumulator as down-counter is shown in Fig. 4.

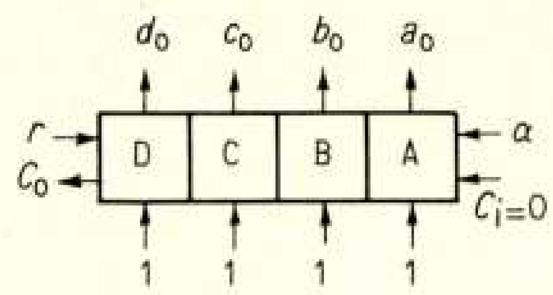


Fig. 4. Binary down-counter.

2.2. *N-Tuple Counter*

If the accumulator is primed with an arbitrary number N, it will operate as N-tuple counter. The priming of the accumulator as '3-tuple' (= triple) counter is shown in Fig. 5. When using

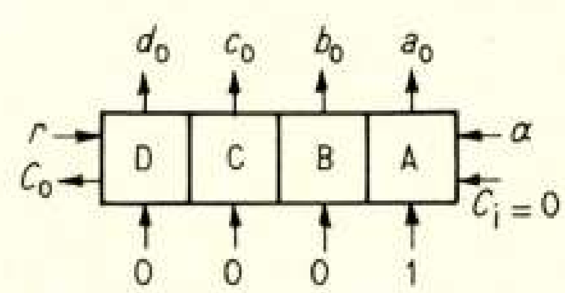


Fig. 5. N-tuple counter, for $N=3$.

the 'two's complement' of N as priming number, the N-tuple counter operates as down-counter.

If the accumulator has p sections, it will run through a com-

plete cycle of states in 2^p counting pulses. However, only the first $(1/N) \cdot 2^p$ counting pulses will be registered in the binary equivalent of N times the number of pulses received. It is not difficult to make the N-tuple counter recycle at the end of this limited series of counting pulses.

2.3. *Shift Register*

The accumulator of Fig. 1 can also be adapted to operate as a shift register. The information signals that are to be shifted through this register will have to be connected to 'carry input' C_i, the output of each section must be connected to its own input. The accumulator provided with these external connections will operate under control of clock pulse α as a left-shift register.

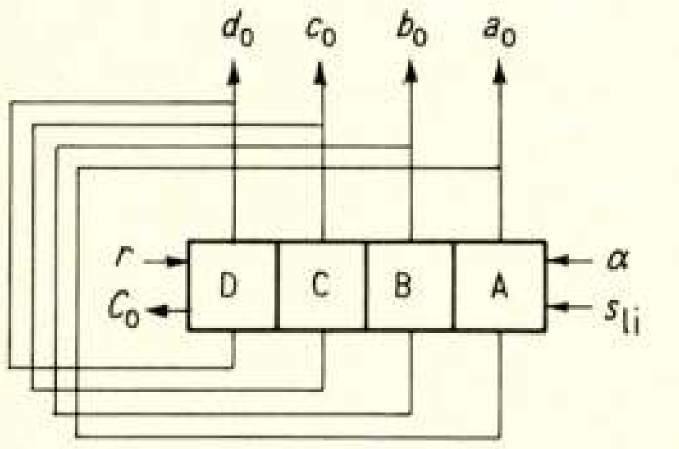


Fig. 6. Left-shift register.

In Fig. 6 the external connections are shown which are required to make the accumulator operate as a left-shift register. It is rather easy to explain this shift action of the accumulator. It is in fact a 'multiply-by-two' operation on the number already taken in the accumulator, or, in other words it is an addition of that number to itself.

The external connections can also be rearranged so that a right-shift register is obtained. For this purpose the complemented output signal of a section has to be fed to the input of the preceding accumulator section. If the complemented outputs are not available, extra inverters have to be inserted in the output-input connections as shown in Fig. 7. In the right-

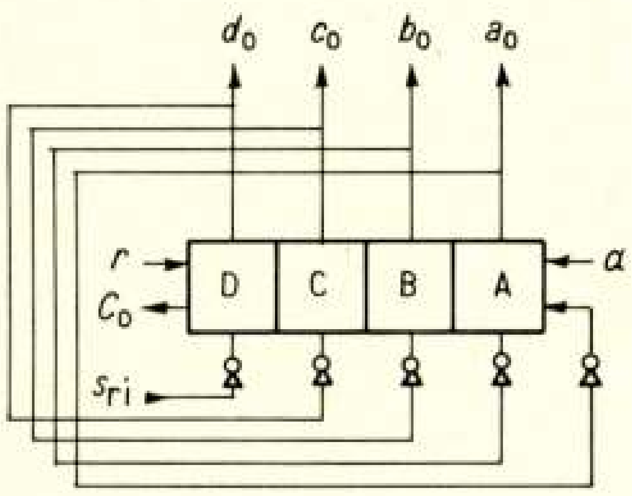


Fig. 7. Right-shift register.

shift process new information is fed to the input of the utmost left-hand section, but in complemented form. This means that the one's complement of the number, stored in the accumulator, is added one digit place further to the right. The adding of the a_o' signal to 'carry input' C_i has the double effect that the digit in the utmost right-hand section of the shift register does not play a role in the adding process and the 1 needed for the always required 'end-around carry' is added automatically. In fact, half of the number stored in the accumulator is subtracted from it.

Example

C_i	=	0	1
(content SR)$_n$	=	0 1 0 1 1 0 1	0 1 0 1 1 0 0
One's complement	=	1 0 1 0 0 1	1 0 1 0 0 1
		+	+
(content SR)$_{n+1}$	=	0 1 0 1 1 0	0 1 0 1 1 0

It is remarked that the change in operating direction of both binary counter and shift register is obtained without change in the direction of the internal signal flow between the sections of the accumulator.

2.4. *Shift-Around Registers*

In many applications of shift registers some content must be shifted around. This type of operation can be obtained both in left-shift and in right-shift registers by connecting the information input to the information output.

In case of a 'left-shift-around' operation carry output C_o of the last section has to be connected to carry input C_i of the first section. In case of a 'right-shift-around' operation output a_o of the first section must be connected to input s_{ri}. For both types of operation several accumulators can be cascaded, when needed.

2.5. *Complementing of Shift Register Content*

When both outputs of each section of a shift register are available, there will be practically no need for complementing the shift register content. However, when these outputs are *not* available, the content of an accumulator can be complemented in the following way.

The 'one's complement' N' of a binary number N, consisting of n digits, is related to N by the following equation.

$$N' + N + 1 = 0 \pmod{2^n} \tag{1}$$

Hence N' can be obtained from N by using the adding operation only:

$$N' = N' + N' + N + 1 = 2N' + N + 1 \tag{2}$$

In this equation N is the given shift register content. '$2N'$ to be added to N' can be obtained by connecting the complemented output of each section to the input of the next higher order section, as is shown in Fig. 8. Finally a 1 has to be added to input a_i of the first section.

Example

$N \;\;= 1\ 1\ 0\ 1$
$2N' = 0\ 1\ 0\ -$
$1 \;\;\;= -\ -\ -\ 1$
$\qquad\qquad +$
$N' \;\;= 0\ 0\ 1\ 0 \pmod{2^n}$

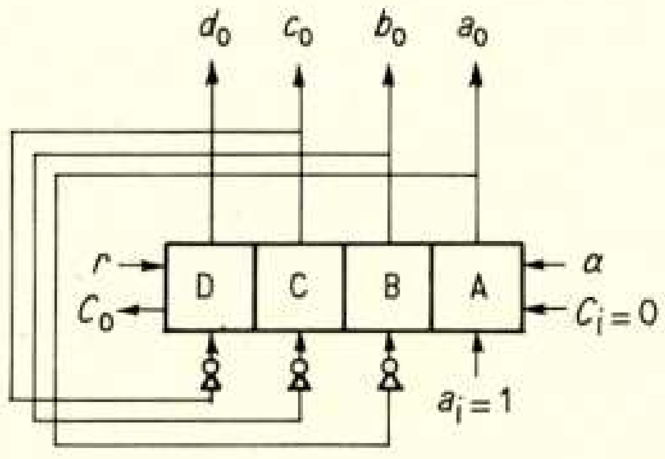


Fig. 8. Complementer.

2.6. *Practical Considerations*

In the preceding sections a number of features have been discussed which can be performed by an accumulator. However, one feature has to be performed at a time, because in the circuit of Fig. 1 the various modes of operation can only be obtained by differing external connections. The usefulness of the accumulator could be greatly increased if its mode of operation could be controlled by a group of program signals.

The number of features or the number of program signals to be incorporated in an accumulator, the number of section to be mounted in one package and last but not least the kind of the features constitute a practical problem with many possible solutions. This problem will not be discussed in this paper. However, one of the possible solutions will be given here, as, in the opinion of the author it is a good compromise between the size of the package and the number of features.

Recently an integrated circuit in dual-in-line package with 18 terminals has appeared on the market as an extention of the line of 14-terminal and 16-terminal packages. It is noted that the already existing 24-terminal packages have much larger dimensions than the 14 ... 18-terminal line.

The 14 ... 18-terminal size of package opens the way for a practical 4-bit accumulator with 8 operating modes. The following modes have been chosen: '*clear*' or simultaneous reset-to-zero of all accumulator sections; '*load*' or simultaneous preset of all sections; '*add*'; '*subtract*'; '*count-up*' in synchronous binary mode; '*count-down*'; and of course '*shift-left*' and '*shift-right*'.

In all types of digital switching circuits these operating modes are very frequently used so that a 4-bit accumulator programmable for these features will have the character of a general-purpose circuit which can replace a number of different types of special-purpose I.C.'s.

The 4-bit programmable accumulator, together with the quadruple standard gate already described in [1], open the way for building many kinds of digital circuits by using only two different types of I.C.'s.

2.7. *Accumulator Diagram*

In Fig. 9 a possible arrangement is shown of an accumulator section in terms of standard gates. It is noted that the number

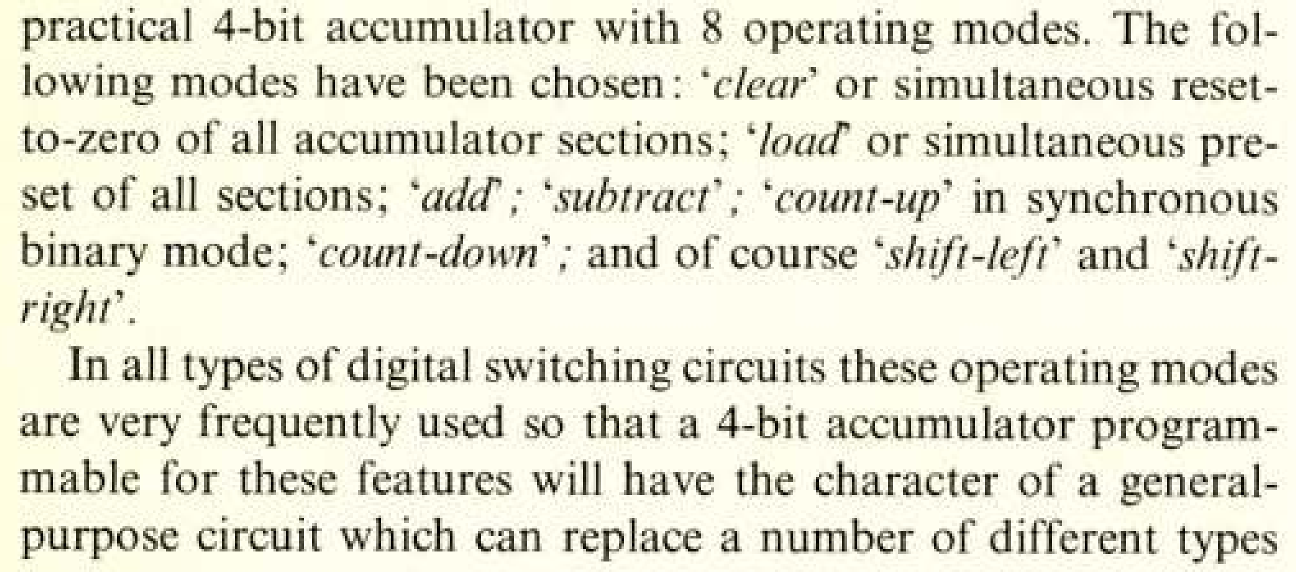


Fig. 9. Accumulator section.

of standard gates is not a direct measure for the actual number of transistors involved in this circuit, and hence not a measure for the complexity of integration, for in most standard gates neither input adapters nor output totem-pole circuits are required. The standard gate symbol has been used to allow for an easy explanation of the various operating modes of the accumulator. In the appendix the actual circuit will be discussed.

The eight operating modes of the accumulator which can be obtained by the use of three program signals p, q, and r, are given in Table 1. These program signals control standard gates SG_1 to SG_5 inclusive in the circuit of Fig. 9.

Table 1.

r	q	p	Mode
0	0	0	subtract
0	0	1	count ↑
0	1	0	count ↓
0	1	1	add
1	0	0	clear
1	0	1	shift →
1	1	0	shift ←
1	1	1	load

In the accumulator arrangement of Fig. 9 the first four operations of Table 1 are performed with the accumulator operating as such. In these four operating modes program signal $r = 0$. The accumulator then consists of: input standard gate SG_1, operating as 'true-complement, zero-one' element; full adder FA consisting of standard gates $SG_{6\ldots8}$; standard gate SG_5 operating as 2-line-to-1-line selector, and D flip-flop DF consisting of standard gates SG_{9-10}.

Standard gate SG_1 provides by its function as true-complement, zero-one element the first four operating modes of Table 1, viz. '*subtract*', '*count* ↑', '*count* ↓' and '*add*'. The full adder FA consists of two 2-input exclusive-OR gates (forming the sum $a_i' \oplus a_0 \oplus c_i' = S$ in case of add condition), and a carry-forming gate SG_8. The position of the 2-line-to-1-line selector SG_5 is determined by program signal r. With $r = 0$ the output of the full adder is connected in true form to the input of the D flip-flop. With $r = 1$ the 4-line-to-1-line selector consisting of standard gates $SG_{2\ldots4}$ is connected to the input of the D flip-flop via the complementing path of gate SG_5. Program signals p and q determine the position of selector $SG_{2\ldots4}$ such, that the last four operating modes of Table 1 are obtained. With program signals $p = 0$ and $q = 0$ the top terminal of gate SG_4 is connected via a complementing path with the input of flip-flop DF so that the 1 input signal gives a clocked reset-to-0 of the circuit.

With program signals $p = 1$ and $q = 1$ the bottom terminal of gate SG_3 is connected with the input of flip-flop DF via a complementing path, so that input signal a_i which is complemented in gate SG_1 under the control of the same pair of input signals, is fed in true form to flip-flop DF. This is the 'load' mode of operation, indicated in Table 1.

With signals $p = 1$, $q = 0$ and $p = 0$, $q = 1$ right-shift input s_{ri} and left-shift input s_{li} are connected with flip-flop DF. In the circuit configuration of Fig. 9 this is the easiest way for obtaining the shift operation. It is rather difficult to provide a shift operation which is derived from the actual accumulator operation. With the direct control of flip-flop DF a higher shifting speed can be obtained.

It is noted that the accumulator can be loaded with new data without a preceding reset-to-zero. It is furthermore noted that

the number of standard gates in the block diagram of Fig. 9 is not a direct measure for the complexity of the accumulator, i.e. the number of transistors involved in this circuit, because in most standard gates neither input adaptors nor output totem-pole circuits are required. The standard gate symbol has been used in Fig. 9 to indicate a logical operation, as a means to explain easily the various operating modes of the accumulator.

A design has been made of a possible actual circuit, that does not differ basically from the circuit given in Fig. 9, and that could be manufactured in TTL integrated circuit techniques.

This design has shown that integration in one chip is feasible. However, to increase its attractivity the design has been made in such a way that the circuit can be split up into two identical parts, each filling a chip. A 4-bit accumulator then contains two chips in one package in the same way as can be found already in the fout-bit adder in TTL technique.

3. Applications

In the preceding sections of this paper nothing more was shown than the fact that the four-bit accumulator in its programmable form has some interesting features, and it was stated that its detailed circuit lends itself for medium-scale TTL integration. The proof of this statement can follow from a discussion of the detailed circuit. However, to save space, we will refrain from that in the present paper.

Presently, a number of applications of the 4-bit programmable accumulator will be given and discussed. Some of these are circuits in which mainly accumulators are used. They will replace already existing integrated circuits. Other ones are applications in which quadruple standard gates and 4-bit programmable accumulators are used. The range of applications shown is limited solely by the allowed size of this paper.

In the following text no further examples will be given of the use of the 4-bit accumulator in one of its eight modes of operation. These operating modes are considered to be sufficiently clear now.

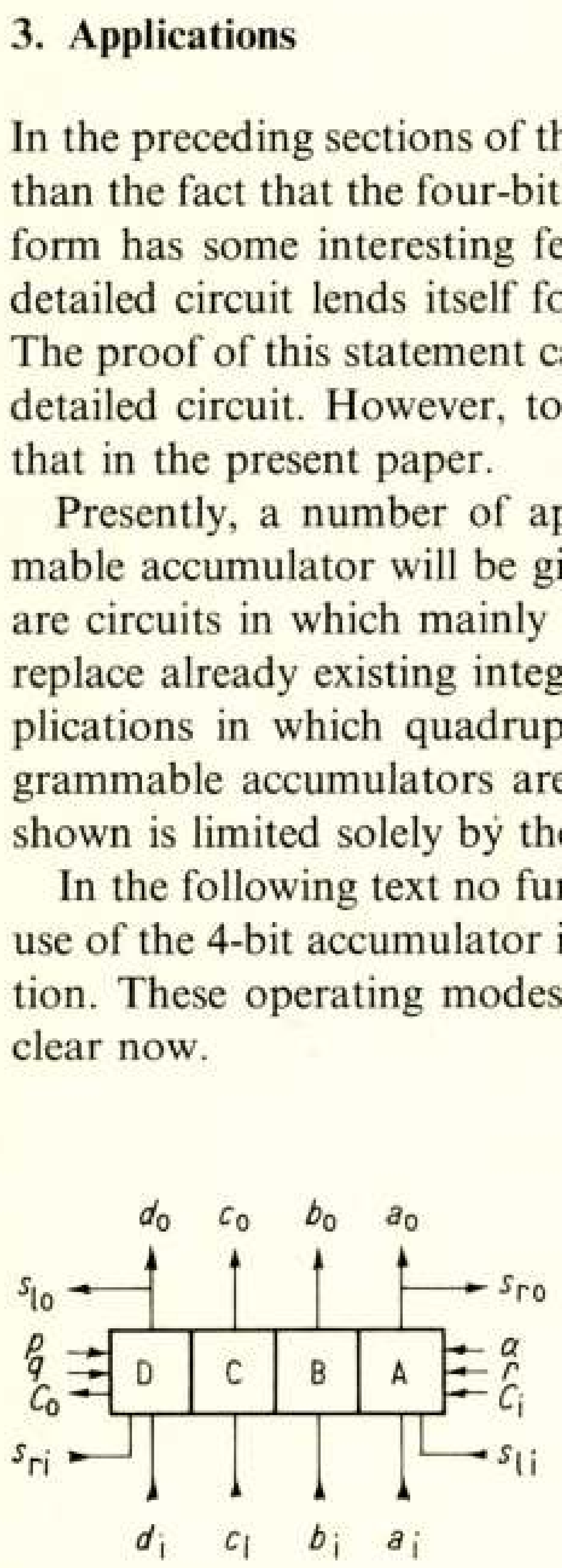


Fig. 10. Accumulator symbol.

The accumulator symbol used in the following diagrams is shown in Fig. 10. It differs from the block symbol given in Fig. 2 in so far that program inputs p, q, and r have been indicated.

3.1. *Divide-by-N Counters*

In many counter applications as e.g. the BCD (Binary Coded Decimal) counter, the natural binary counting cycle of 2^n (n is the number of counting sections) has to be shortened to a smaller number. In a four-section BCD counter the natural cycle length of 16 has to be shortened to 10, which means that on the tenth pulse of each cycle, the counter has to be reset to zero. This recycling can always be performed by means of some extra gates on any desired number of pulses received.

In the four-section counting circuit shown in Fig. 11 the number of pulses of the counting cycle can be adjusted by means of a plug board. This circuit consists of an accumulator, programmed to count up, and of standard gates $SG_{1 \ldots 5}$. The outputs of the counter sections control via the plug board a four-wide NAND gate, consisting of standard gates $SG_{1 \ldots 3}$ followed by standard gate SG_4 programmed as inverter.

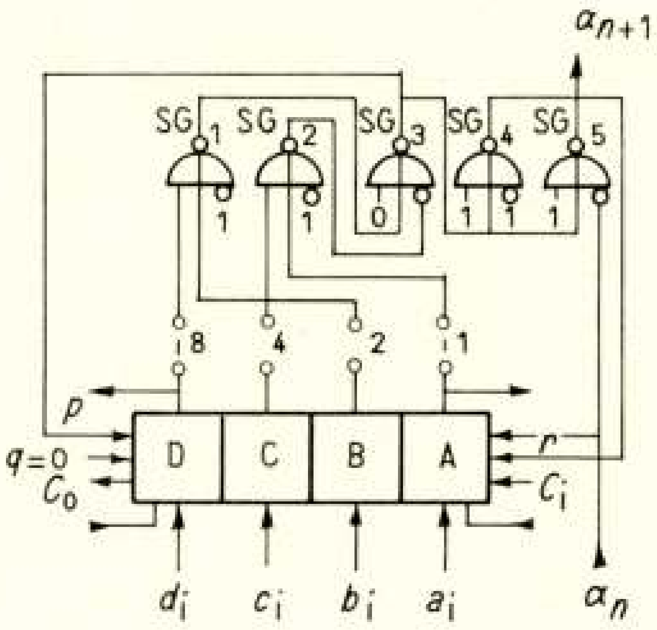


Fig. 11. Divide-by-N counter.

The plug board in Fig. 11 is connected such that the counter will operate as a decade counter (BCD counter). The dotted line connections make the output of gate SG_3 change from 1 to 0 at the end of the ninth counting pulse α.

During the counting cycle (the first nine pulses) program code $pqr = 100$. After the ninth counting pulse this program code must be changed into the code for 'clear', $pqr = 001$, in order to obtain a reset-to-zero on the tenth counting pulse of the cycle. This means that the output of gate SG_3 can control the p program signal and that the output of gate SG_4 generates the r program signal. Program signal q is 0 during this counting process.

It is noted that in this counting procedure the 'load' mode can also be used instead of the 'clear' mode, but this necessitates all section inputs to be kept at the 0 logical level, which blocks the possibility of presetting the counter.

Gate SG_5 in the decade-counter generates a counting pulse (α_{n+1}) for the next decade at each tenth pulse of a counting cycle. In this type of counter all sections of one decade are controlled synchronously.

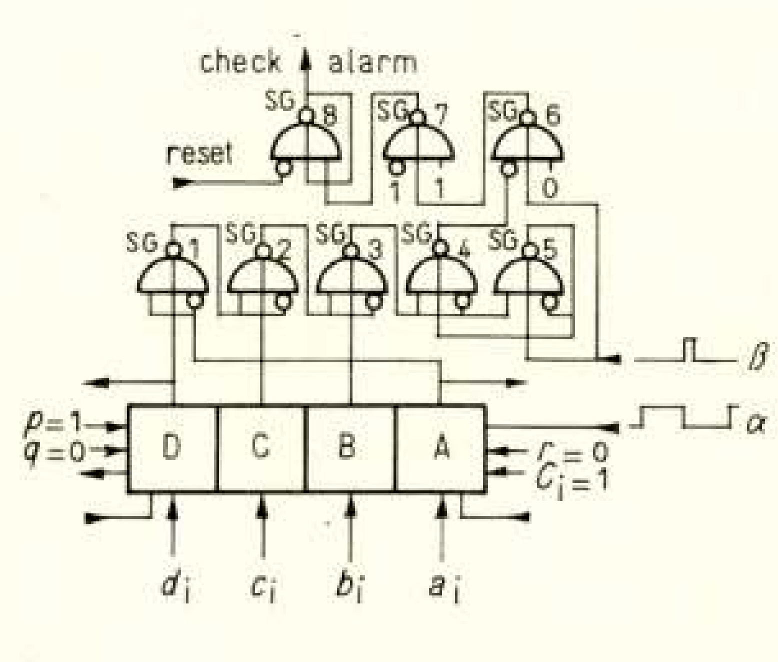


Fig. 12. Self-checking binary counter, circuit diagram.

3.2. *Self-Checking Binary Counter*

When using a single binary counter on long series of counting pulses one is never sure that the counting result is correct. It is always possible that interference from an electromagnetic field reverses the state of one or more sections of the counter at an arbitrary moment. A great percentage of these interferences will affect one section only. The percentage of cases that two or more sections are reversed by an interference is very small, therefore a parity check on the operation of a binary counter may be worthwhile.

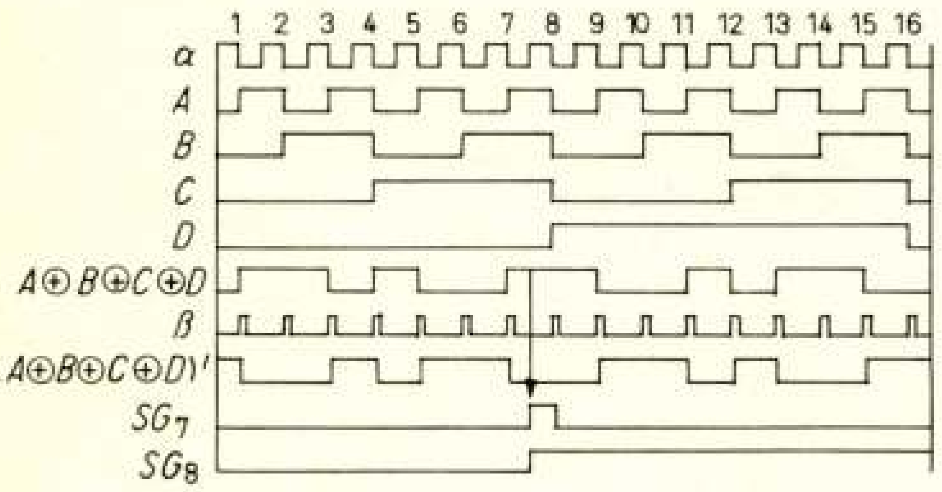


Fig. 13. Self-checking binary counter, time chart.

In Fig. 12 the circuit is given of a self-checking binary counter; in Fig. 13 its time chart is shown. The counter used in the circuit of Fig. 12 is an accumulator, programmed with $pqr = 100$ as up-counter. Standard gates SG_1 to SG_3 are connected as exclusive-OR gates. They form the 'mod-2'-sum of the states of counter sections A, B, C, and D. Directly after the state change of counting pulse α from 1 to 0, a sensing pulse β of short duration is given.

Standard gate SG_5 is connected as a latch clocked by sensing pulse β. In this latch $(A \oplus B \oplus C \oplus D)'$ is stored. The outputs of gates SG_3 and SG_5 are fed to the exclusive-OR gate formed by gate SG_4. In this gate sum $(A \oplus B \oplus C \oplus D) \oplus (A \oplus B \oplus C \oplus D)'$ is formed, which sum is 'one' as long as no interference occurs.

If, for example an interference occurs during the seventh pulse of a cycle of 16 (see arrow in Fig. 13), the output signal of gate SG_4 will immediately change from 0 to 1 and, sensing pulse β being 0, the output of gate SG_7 will follow. This forms the set signal for gate SG_8 which is connected as set-reset trigger. The output of SG_8 changes from 0 to 1 which is the check alarm signal. This alarm signal will last until trigger SG_8 is reset.

3.3. *BQ Decade Counter*

The 4-bit programmable accumulator can be used – without auxiliary hardware – as a biquinary-coded decade counter. The external connections required to obtain this operation are shown in Fig. 14. In this circuit the accumulator is programmed with $pqr = 110$ ('add'). Under control of counting pulses α_1 the circuit will run through a cycle of 10 combinations of the biquinary code.

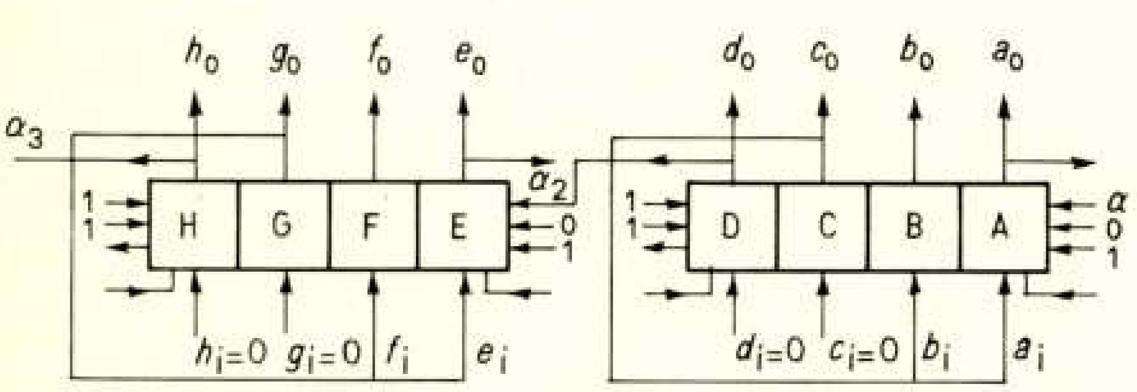


Fig. 14. BQ decade counter.

In the circuit of Fig. 14 this sequence of code combinations is obtained by connecting output c_o with inputs a_i and b_i so that after code combination 0100 the content of the accumulator will be increased with $3 + 1 = 4$ instead of 1. The counting code value of section D will then be five.

The same external connection makes that on the tenth counting pulse of each cycle the counter will jump from code combination number 9 (1100) to 0000. During the jump an outgoing carry will be produced (α_2) which can be used for the control of the next decade.

3.4. *Synchronous Binary Rate Multiplier*

In the examples given so far the accumulator was more or less a replacement of a special-purpose I.C. Perhaps due to the fact that in some examples the 'load' mode of operation was used for special shift features, no extraordinary accumulator operation was obtained.

An exception to that is the synchronous binary rate multiplier. This type of circuit is available as special-purpose I.C. in a 6-bit chip. It contains a synchronous binary counter and a programmed group of n output gates so connected that, when characterized by an n-bit word ($uvwxyz$), the number N of output pulses in a cycle of 2^n input pulses is:

$$N = u \cdot 2^5 + v \cdot 2^4 + w \cdot 2^3 + x \cdot 2^2 + y \cdot 2^1 + z \cdot 2^0 \qquad (3)$$

for $n = 6$.

However, the number of output pulses in this type of circuit is, with regard to the desired rate, not distributed in the best possible way during the cycle of 2^n pulses, because the pulses missing in the regular sequence of 2^n pulses are not removed at the best possible places.

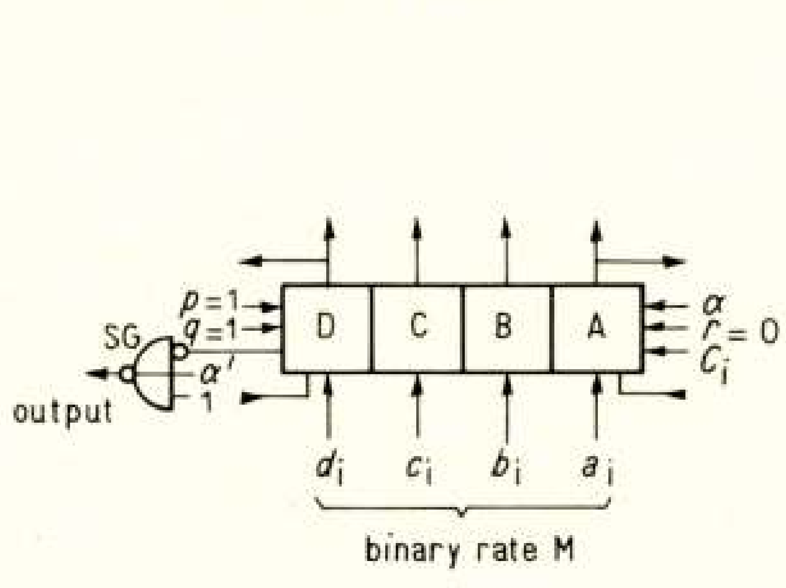


Fig. 15. Synchronous binary rate multiplier.

This difficulty can entirely be overcome by using the binary rate multiplier circuit of Fig. 15. In this circuit the accumulator operates in the 'add' mode with program $pqr = 110$. The desired binary rate is fed to input terminals a_i, b_i, c_i, and d_i. On each pulse α this number is added to the content of the accumulator. The accumulator operates in fact as an N-tuple counter, N being the desired binary rate. In this type of operation the carry output C_o of the accumulator generates a series of zeros and ones, in accordance with the desired binary rate.

Table 2 shows the accumulator contents and carry output C_o during a counting cycle.

Table 2.

C_o	D	C	B	A	No.
0	1	0	1	1	1
1	0	1	1	0	2
1	0	0	0	1	3
0	1	1	0	0	4
1	0	1	1	1	5
1	0	0	1	0	6
0	1	1	0	1	7
1	1	0	0	0	8
1	0	0	1	1	9
0	1	1	1	0	10
1	1	0	0	1	11
1	0	1	0	0	12
0	1	1	1	1	13
1	1	0	1	0	14
1	0	1	0	1	15
1	0	0	0	0	16

Binary rate multipliers with a larger number of sections can be obtained by cascading the desired number of accumulators.

3.5. *Squaring Pulse Counter*

The square of the number of pulses in a pulse series can be determined in a synchronous way together with the reception of that pulse series. The distances d between successive perfect squares form an arithmetical series with ratio 2 and starting from 1. It is easy to form the terms of that series by means of a binary counter; however, to get the series started from 1 the weights of the successive sections of that counter must be 2, 4, 8, etc. The missing 1 can be added automatically, as will be explained in the following.

The square of an integer number $(p + 1)$ is determined by the following equation.

$$s_{n+1} = (p + 1)^2 = p^2 + 2p + 1 = s_n + 2p + 1 \qquad (4)$$

Table 3 illustrates the implementation of this equation.

Table 3.

p	0	1	2	3	4	5	6	7	8	9	10
BC_n	0	2	4	6	8	10	12	14	16	18	20
CA_n	1	1	1	1	1	1	1	1	1	1	1
AC_n	0	1	4	9	16	25	36	49	64	81	100
AC_{n+1}	1	4	9	16	25	36	49	64	81	100	

In this table $BC_n = 2p$, $CA_n = 1$, $AC_n = s_n$ and $AC_{n+1} = s_{n+1}$.

The circuit of a squaring pulse counter is given in Fig. 16. It consists of an accumulator AC operating as such with program code pqr = 110 and of another accumulator operating as binary up-counter with program code pqr = 100. The double of the number of pulses received ($2p$) is added to the content of accumulator AC under control of each received counting pulse α. For this purpose the counter outputs are connected with the accumulator inputs of the double weight. The missing 1 mentioned above, can be added as a 1 permanently fed to the input of section Z of accumulator AC.

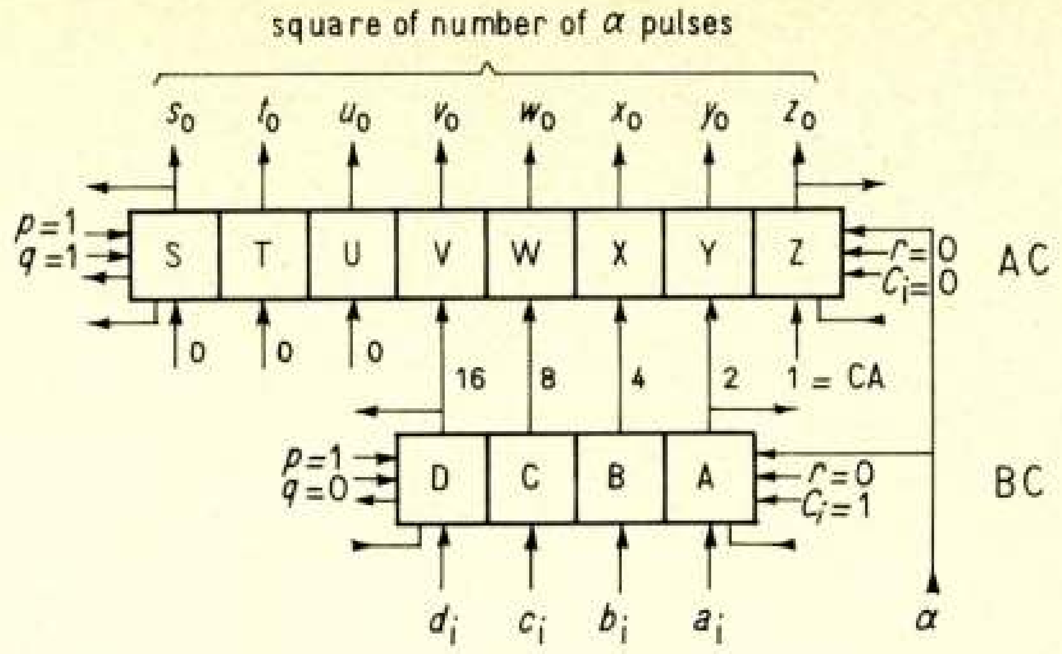


Fig. 16. Squaring pulse counter.

3.6. *Reflected and Natural Binary Counter*

Another counter which can elegantly be designed using accumulators is the combined reflected and natural binary counter.

A four-section combined reflected and natural binary counter as shown in Fig. 17 can be designed using two 4-bit accu-

Table 4.

W	D	X	C	Y	B	Z	A	No.
15	8	±7	4	±3	2	±1	1	
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1
0	0	0	0	1	1	1	0	2
0	0	0	0	1	1	0	1	3
0	0	1	1	1	0	0	0	4
0	0	1	1	1	0	1	1	5
0	0	1	1	0	1	1	0	6
0	0	1	1	0	1	0	1	7
1	1	1	0	0	0	0	0	8
1	1	1	0	0	0	1	1	9
1	1	1	0	1	1	1	0	10
1	1	1	0	1	1	0	1	11
1	1	0	1	1	0	0	0	12
1	1	0	1	1	0	1	1	13
1	1	0	1	0	1	1	0	14
1	1	0	1	0	1	0	1	15

Table 5.

Y	B	Z	A
jump addend			state
0	0	0	0
0	0	1	1
0	0	1	1
1	0	1	1
1	1	1	0
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	0	1	1
1	0	1	1
1	0	1	1
0	1	1	0
1	1	1	1
0	1	0	1
1	0	1	1

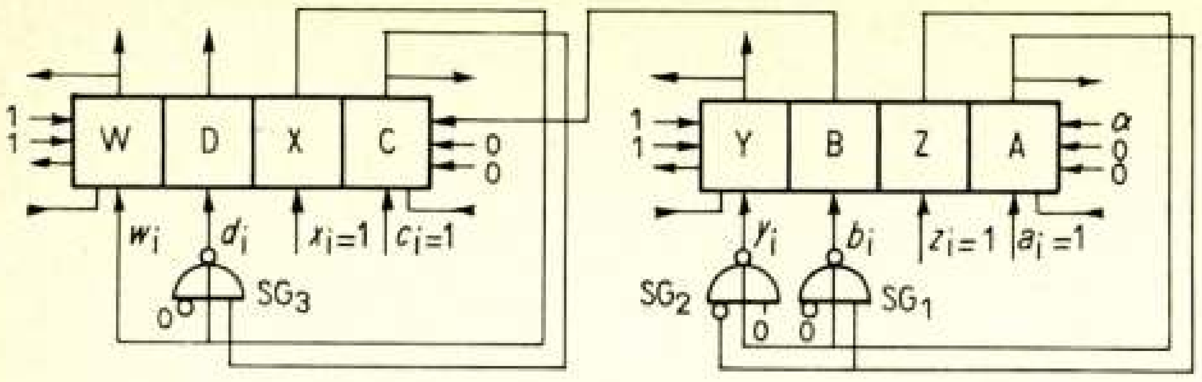


Fig. 17. Natural and reflected binary counter.

mulators. In the diagram of Fig. 17 sections ABCD operate in the natural binary code, and sections WXYZ operate in the reflected binary code.

The 4-bit accumulators in the circuit are programmed with code $pqr = 110$ to operate in the 'add' mode, i.e. to operate as an accumulator. In Table 4 is shown how the sections of the combined counter operate.

In Table 5 is shown which numbers have to be added to the various states of accumulator YBZA in order to obtain the operation defined in Table 4.

It follows from Table 5 that sections Z and A permanently must be fed with (1 1) so that these sections actually operate as a down-counter, which in fact also follows from Table 4. The jump addend following from Table 5 for sections Y and B is $y_i = a_o + z_o$ and $b_i = a_o' \cdot z_o$. Both functions can be implemented easily by means of standard gates, as is shown in the circuit of Fig. 17.

Sections W, D, X, and C of the second half of the counter can be controlled by signal b_o, i.e. the output signal of section B. The last section of the last 4-bit accumulator, in this case section W, has to be controlled differently from other last sections in order to prevent that the normal counting cycle as given in Table 4 is followed by a reverse counting cycle. The input signal of section W simplifies to $W_i = x_o$.

3.7. *Square-Root N_α Counter*

The programmable accumulator is well suited to determine the square root of the number N of counting pulse series α during its reception. The circuit is shown in Fig. 18. It consists mainly of an accumulator A ... H fitted here with 8 sections and



Fig. 18. Square-root N_α counter.

programmed with $pqr = 110$, to operate in its 'add' mode, and of a second accumulator S ... Z programmed to operate either in its 'count-down' mode ($pqr = 010$) or in its 'load' mode ($pqr = 111$). Sections B ... H inclusive can feed sections Z ... T inclusive with fresh information when OR-gate OG senses accumulator S ... Z to be in its 0 state. OR-gate OG, via inverter SG_1, controls program signals p and r of accumulator S ... Z. In the 0-state of accumulator S ... Z output of OR-gate OG is 0, so that program code $pqr = 111$ (*load*). The contents of accumulator B ... H is fed into accumulator Z ... S at the next counting pulse α. This changes the output of gate OG from 0 to 1 and that of inverter SG_1 from 1 to 0, so that the program code of accumulator S ... Z becomes $pqr = 010$ (*count-down*).

Accumulator A ... H is used with program code $pqr = 110$ in its 'add'-mode, but, because $C_i = 1$, $b_i \ldots h_i = 0$, and input $a_i = 1$ only in the start condition, this accumulator, except for the first step, will operate as an up-counter.

Standard gate SG_3, fitted out as set-reset trigger, feeds in the start condition of the circuit a '1' to input a_i, so that on reception of the first counting pulse α (all accumulators in the 0 state!) $c_i = 1$ and $a_i = 1$ are added and stored directly in section B as a 1. Trigger SG_3 is reset to 0 by the first counting pulse α and remains in that 0 state during the whole counting process.

It is noted that gate SG_2 is open to feed a counting pulse α to accumulator A ... H only when accumulator S ... Z is in its all-zero state.

During the first counting step accumulator S ... Z remains in its 0 state because initially accumulator A ... H was also in the 0 state. On reception of the second counting pulse α, sections Z and A are set to 1 and gate SG_2 is blocked.

Accumulator S ... Z, operating as down-counter, is back in its all-zero state after one counting pulse α (the third) so that on the fourth α-pulse section Z is again set to 1 while section C is also set to 1 and sections A and B are reset to 0.

The following counting process will develop itself. It is sufficient to show this counting process only for sections X ... Z and A ... D.

X	Y	Z	α	X	Y	Z	α	X	Y	Z	α	X	Y	Z	α
0	0	0	0	0	0	0	1	0	0	1	2	0	0	0	3
D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	0	0.	0	0	0	1.	0	0	0	1.	1	0	0	1.	1

X	Y	Z	α	X	Y	Z	α	X	Y	Z	α	X	Y	Z	α
0	0	1	4	0	0	0	5	0	1	0	6	0	0	1	7
D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	1	0.	0	0	1	0.	0	0	1	0.	1	0	1	0.	1

X	Y	Z	α	X	Y	Z	α	X	Y	Z	α	X	Y	Z	α
0	0	0	8	0	1	0	9	0	0	1	10	0	0	0	11
D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	1	0.	1	0	1	1.	0	0	1	1.	0	0	1	1.	0

X	Y	Z	α	X	Y	Z	α	X	Y	Z	α	X	Y	Z	α
0	1	1	12	0	1	0	13	0	0	1	14	0	0	0	15
D	C	B	A	D	C	B	A	D	C	B	A	D	C	B	A
0	1	1.	1	0	1	1.	1	0	1	1.	1	0	1	1.	1

X	Y	Z	α
0	1	1	16
D	C	B	A
1	0	0.	0

In this root-extracting process the root of an integer number of counting pulses is determined with an accuracy of $\frac{1}{2}$ indicated by section A. The resulting roots are accurate at all integer numbers N which are perfect squares, and nearly accurate at all integer numbers $N + \sqrt{N}$. The resulting root is found in accumulator A ... H.

3.8. *Arithmetic Unit*

The part of this paper dealing with applications will now be concluded by an example of the possible use of the accumulator in an arithmetic unit. More specifically, the implementation of *division* and *multiplication* algorisms by means of accumulators will be shown. The implementation of *addition* and *subtraction* will not be discussed explicitly because these operations form the basis of the accumulator.

In the implementation of both division and multiplication algorisms the accumulator operations 'add' or 'subtract', and at the same time 'shift-right' or 'shift-left' are required. If the multiplicand, resp. the divisor are stored in a non-shift register, simultaneous 'add' and 'shift' operations must be incorporated in one accumulator. This leads to an arithmetic unit with a minimum of length of various registers. However, this combined operating mode is not available in the 4-bit standard accumulator discussed above. Introduction of the combined 'add-shift' operating mode with all its necessary variations will increase the number of terminals of the accumulator package from 18 to 22. For such an accumulator the large standard 24-terminal package would have to be preferred. This is against the desire of having the same size of package for the quadruple standard gate and the 4-bit standard accumulator.

It will be shown that, at the cost of some extra hardware, very versatile division and multiplication circuits can be designed using the standard accumulator. From the point of view of operating time, these circuits are compatible with the best possible circuits now available.

3.8.1. *Division*

In the diagram of Fig. 19, a circuit of an arithmetic unit is given which is designed to perform the non-restoring division of positive binary numbers. Each of the three accumulators shown ($SA_{1 \dots 3}$) has four sections. However, their length can be increased to any desired number of bits.

The division process in the arithmetic unit of Fig. 19 consists of three phases:

1. *Clearing of accumulator* SA_1, *loading of accumulator* SA_2 with the divisor (lowest-valued bit in section V), and *loading of accumulator* SA_3 with the dividend (lowest-valued bit in section Z).

Accumulator SA_1 controlled by $p_1 = 0$ will operate in accordance with program code $p_1q_1r_1 = 001$, which is the code for 'clear'. The 'clear' operation will be performed on an α clock-pulse.

Accumulator SA_2 controlled by $k = 0$ via gates SG_9 and SG_{10} will operate in accordance with program code $p_2q_2r_2 = 111$ ('load'). Control signal $k = 0$ positions gates $SG_{1 \dots 4}$ so that input terminals a_i ... d_i are connected with accumulator SA_2. This accumulator then will be loaded on clock pulse β.

Accumulator SA_3 controlled by $r_3 = 1$ gives a program code $p_3q_3r_3 = 111$ ('load') via inverter SG_{11} and gate SG_{14}. Output signal $r_3' = 0$, of gate SG_{11}, positions gates $SG_{5 \dots 8}$ so that input terminals a_i ... d_i are connected with accumulator SA_3 and opens gate SG_{12} for clock pulse γ so that the load operation can be performed.

2. *Alignment of divisor and dividend.* To perform this operation control signal p_1 of accumulator SA_1 is changed from 0 to 1 so that its program code $p_1q_1r_1 = 101$, which is the code for 'right-shift'.

Control signal k of accumulator SA_2 is changed to 1 so that the program code of this accumulator will be determined by the output of set-reset trigger SG_{15}. At the same time this trigger is set to 1 by control pulse m. This changes the program code of accumulator SA_2 into $p_2q_2r_2 = 101$ (right-shift'). Output signal 1 of trigger SG_{15} keeps gate SG_{16} closed, and this signal, when complemented by inverter SG_{17}, opens gate SG_{18}. Simultaneously generated clock-pulses α and β now shift zeros into accumulator SA_2 and shift the divisor stored in accumulator SA_2 into accumulator SA_1. At the start of the alignment process, control signal r_3 of accumulator SA_3 is changed

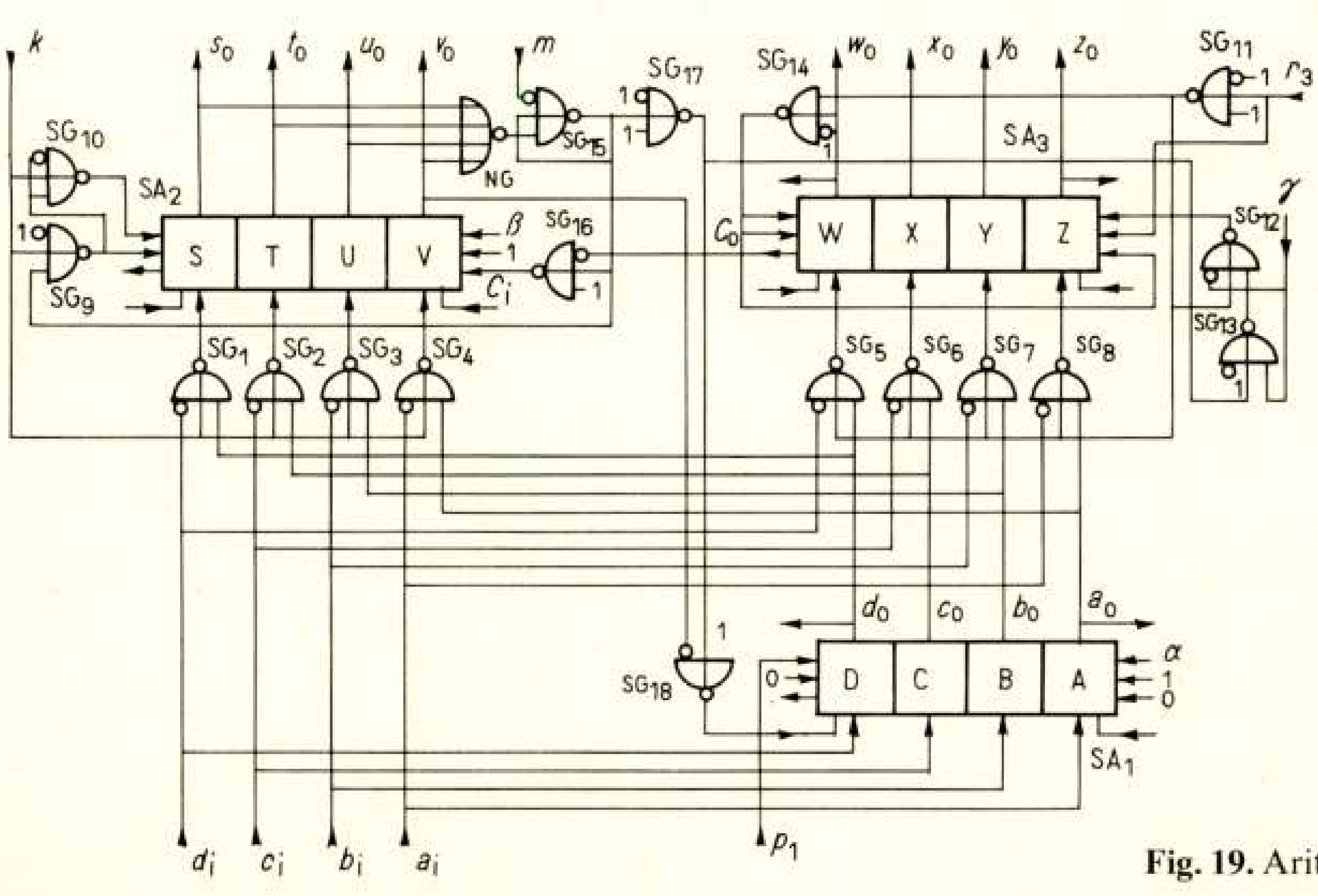


Fig. 19. Arithmetic unit for non-restoring division of positive numbers.

to 0, so that during the right-shift operation of accumulators SA_1 and SA_2, clock pulses γ applied to accumulator SA_3 via gates SG_{13} and SG_{12} remain inoperative because the output signal of inverter SG_{17} keeps gate SG_{13} closed for clock pulse γ.

Control signal $r_3 = 0$ connects the output of gate SG_{13} via gate SG_{12} with the clock-pulse input of accumulator SA_3. Gate SG_{13} is controlled by the output of inverter SG_{17}, so that at the end of the alignment process when the output of NOR-gate NG becomes 1, clock pulse γ is connected with accumulator SA_3. Furthermore control signal $r_3' = 1$ programs gate SG_{14} to operate as inverter, so that the program code of accumulator SA_3 can either be $p_3q_3r_3 = 110$ ('add') or $p_3q_3r_3 = 000$ ('subtract').

When the divisor in its positive representation, except its 0-sign digit and leading zeros, is shifted out of accumulator SA_2, NOR gate NG generates a '1' output signal which resets trigger SG_{15} to 0.

Output signal 0 of trigger SG_{15} changes the operation code of accumulator SA_2 from $p_2q_2r_2 = 101$ into $p_2q_2r_2 = 011$ ('left-shift'). The program code of accumulator SA_1 remains unchanged.

The output signal of trigger SG_{15} opens gate SG_{16} and closes gate SG_{18}. The output signal of inverter SG_{17} opens gate SG_{13} for clock pulse γ (simultaneously given with α and β) so that the actual division can start.

3. *Execution of the non-restoring division.* The operation of NOR-gate NG as detection of the all-zero state of accumulator SA_1 is the start of the actual division. All accumulators are now programmed for that. However at this moment the divisor is stored in accumulator SA_1 without its 0-sign digit and leading zeros. There is no objection for that in the non-restoring division, because this lack in alignment of divisor and dividend is corrected automatically in the following division process.

Under the control of signal $r_3' = 1$ the divisor stored in accumulator SA_1 can be fed to accumulator SA_3 in which the dividend is stored. It is noted that now the complementary path of gates $SG_{5\ldots8}$ is used, which means that operations 'add' and 'subtract' have to be interchanged.

The operations 'add' and 'subtract' of accumulator SA_3 are, via gate SG_{14}, controlled by the digit in the utmost left-hand section of accumulator SA_3. In this section the sign digit is stored of dividend and all partial remainders. This digit being 0, program code $p_3q_3r_3 = 110$ will be obtained; when this digit is 1, program code $p_3q_3r_3 = 000$ will result. These combinations are respectively the code for 'add' and 'subtract' which are the codes actually to be used.

In the non-restoring division process the quotient digits are generated as carry digits of accumulator SA_3. These digits are shifted via gate SG_{16} into accumulator SA_2, so that at the end of the division process accumulator SA_2 will contain the correctly placed quotient. It is noted that the output of SG_{14} not only controls the 'add' or 'subtract' mode of operation of accumulator SA_3 but also the carry input of this accumulator to provide for the extra one required for the two's complement representation when the divisor has to be subtracted from the dividend. The division is completed after a number of clock pulses, that equals the number of sections of accumulator SA_1 + one.

An advantage of the arithmetic unit shown in Fig. 19 is that it allows for an extreme variation in the length of divisor and dividend. Considered from this point of view the extra hardware in the form of accumulator sections seems justified.

3.8.2. *Multiplication*

In Fig. 20 the interconnections of accumulators $SA_{1\ldots3}$ are given for the execution of the multiplication of positive numbers.

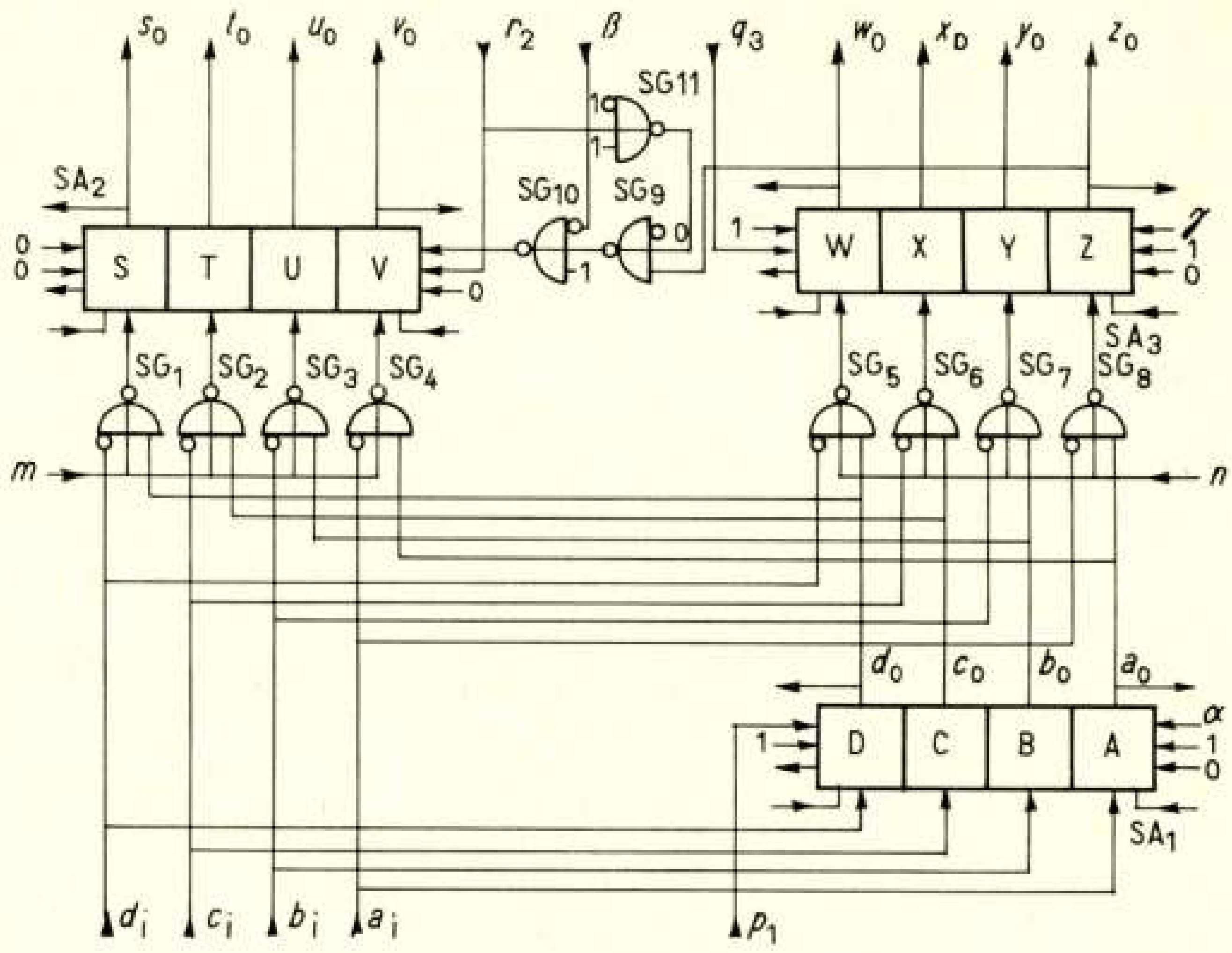


Fig. 20. Arithmetic unit for multiplication of positive numbers.

'Clear' and 'load' operations of accumulators $SA_{1\ldots3}$ are similar to those already described for the division process.

Accumulator SA_1 must be loaded with the multiplicand with a as lowest-valued bit, accumulator SA_2 must be cleared, and accumulator SA_3 must be loaded with the multiplier with z as lowest-valued bit (control signal $n = 0$). During the multiplication process accumulator SA_1 is programmed with $p_1q_1r_1 = 011$ (control signal $p_1 = 0$), so that it will perform a shift-left operation; accumulator SA_2 is programmed with $p_2q_2r_2 = 000$ (control signal $r_2 = 0$) and accumulator SA_3 with $p_3q_3r_3 = 101$ (control signal $q_3 = 0$), so that it will perform a right-shift operation.

Because the multiplicand stored in accumulator SA_1 is fed to accumulator SA_2 via the complementing paths of gates $SG_{1\ldots4}$, accumulator SA_2 must be programmed for 'subtraction' when performing the multiplication process. With program signal $r_2 = 0$ the program code of accumulator SA_2 is, as already stated, $p_2q_2r_2 = 000$, i.e. the code for 'subtract' so that the content of accumulator SA_1 will be added to that of accumulator SA_2 if gate SG_{10} is open for clock pulse β (control signals $m = 1$ and 1 and $n = 0$).

The utmost right-hand bit stored in accumulator SA_3 determines whether or not the multiplicand stored in accumulator SA_1 will be added to the partial product already stored in accumulator SA_2; in other words, whether or not clock pulse β controlling the operation of accumulator SA_2 will be fed to that accumulator via gate SG_{10} or not. This gate is open with $z_0' = 0$.

As a result of each clock-pulse group (α, β, and γ) the multiplicand is shifted over one bit to the left with respect to the already formed partial product, and this multiplicand is added to that partial product under control of the utmost right-hand bit in accumulator SA_3.

Both examples show the usefulness of the accumulator in its application to arithmetic units.

4. Economic Considerations

Finally the applicability of the 4-bit programmable accumulator will be considered from the point of view of prices. The accumulator has about the same degree of complexity as the synchronous up-down 4-bit binary counter of the SN 74191 type. Hence it may be expected that the price of the generally applicable 4-bit accumulator need not exceed that of this binary counter. Compared with other circuits of the same degree of complexity the 4-bit synchronous binary counter SN 74191 shows a relative decrease in price compared with the rapidly decreasing prices of digital I.C.'s for which its usefulness may be credited. In the following it will be assumed that the bulk price of the 4-bit programmable accumulator is the same as that of the 4-bit synchronous binary counter SN 74191.

With regard to price and features a number of special-purpose I.C.'s will now be compared with the 4-bit programmable accumulator. In that comparison the bulk price of the accumulator will be weighed against the single piece price of the special-purpose I.C. By this attention is called to the fact that in many cases the special-purpose I.C.'s will have to be purchased in small numbers.

a. *Binary Rate Multiplier*
The price ratio between the commercially available 6-bit binary rate multiplier and the 4-bit accumulator, which can be operated as a 4-bit binary rate multiplier, is 2.25. Taking the number of bits into account the price ratio is 1.5 in favour of the accumulator.

This leads to the conclusion that the use of the programmable accumulator as binary rate multiplier, which has the best possible distribution of output pulses, is fully justified.

When the accumulator is available, the existence of the now commercially available binary rate multiplier becomes problematic. In this relation the question can also be posed to prospective manufacturers of binary rate multipliers: Why not the accumulator?

b. *Right-shift, Left-shift Registers*
The accumulator, used as shift register, cannot compete in price with the single-function 4-bit shift registers with parallel data input, eventually also with parallel data output. The price ratio is 2 : 1 in favour of the special-purpose I.C.'s.

However, the special-purpose 8-bit parallel-access, left-shift, right-shift register is slightly more expensive than two 4-bit accumulators. Both have the same group of shift features and from this point of view they can be considered equal.

c. *Binary Counters*
The 4-bit accumulator used as binary counter is about three times as expensive as ordinary TTL integrated 4-bit asynchronous counters with no parallel load or up-down count features. But this changes into a ratio of 1.25 in favour of the 4-bit programmable accumulator when these features are also required.

The 4-bit programmable accumulator has still more interesting counting features as e.g. the N-tuple mode of operation so that in all cases where complicated counting problems have to be solved, the 4-bit programmable accumulator will be in the advantage.

The examples given in the preceding text show that many switching problems can be solved by means of accumulators. The size of this paper did not allow to show the equivalent circuit design in commercially available TTL integrated circuits. But generally speaking, a non-programmable 4-bit accumulator can be simulated by means of a 4-bit binary adder and two dual D flip-flops, which cost about the same as the accumulator when mounting costs and space are taken into account.

5. Conclusion

With respect to technical feasibility, applicability and economy there are no reasons to reject manufacturing the quadruple standard gate discussed in the previous paper [1] and the 4-bit programmable accumulator, discussed in the present paper, as TTL I.C. Both circuits can replace a number of already existing TTL I.C.'s purely on a basis of economy and technical features. In addition to this field of application, the two programmable standard circuits open the possibility of designing information-processing circuits applying only two types of TTL I.C.'s. With this kind of circuit design, maintenance can be greatly facilitated resulting in reduced maintenance costs. Furthermore the number of different spare circuits will decrease to a minimum. In order to obtain these important advantages, it may be quite possible that the costs of the standard integrated circuits, to

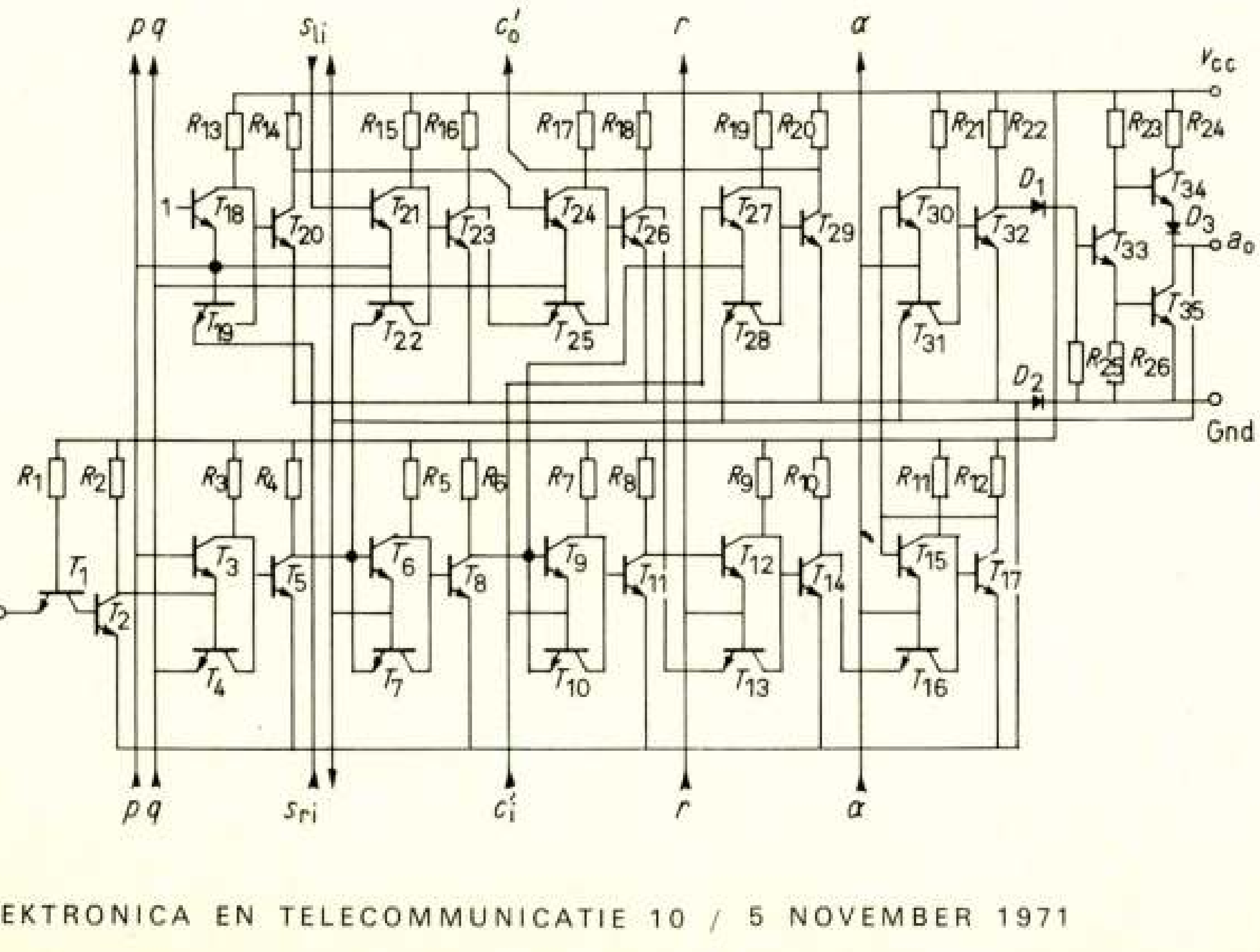


Fig. 21. Accumulator diagram.

some extent being higher than those of the replaced special-purpose types of integrated circuits, must be accepted.

6. Appendix

In Fig. 9 (paragraph 2.7.), a symbolic implementation has been given of a section of an accumulator. In Fig. 21 the design of the actual circuit in TTL logic is shown. It consists of ten standard gate circuits each containing three transistors (transistors $T_{3 \dots 32}$), an input adapter (transistors T_{1-2}) and a totem-pole output circuit (transistors $T_{33 \dots 35}$). The input adapters for program control signals p, q, and r, clock pulse α, carry input c_i, shift-right and shift-left input are not shown in Fig. 21, nor the carry output circuit.

Each of the ten standard gates, together forming one accumulator section, is shown with three transistors, so that for one 4-bit accumulator this pattern can be repeated 40 times. However, not all amplifying transistors are necessary. Transistors T_5, T_{14}, T_{20}, and T_{23} can, if so desired, be omitted. The circuit must then be adapted. The total number of transistors in a 4-bit accumulator then decreases by 16. It is not yet clear what should be preferred; per section 10 equal standard gates with three transistors, or a minimum number of transistors per accumulator section. The circuit of Fig. 21 shows that the 4-bit programmable accumulator as developed in this paper, is of the same degree of complexity as the 4-bit synchronous binary up-down-counter SN 74191, so that from the point of view of possible realization in hardware there will be no objections against this circuit.

References

[1] OBERMAN, R. M. M.: Standard Gates. 'De Ingenieur', Jrg. 83, nr. 27, blz. ET 83. (1971).

[2] RICHARDS, R. K.: 'Arithmetic Operations in Digital Computers'.

[3] Motorola Monitor, Vol. 6 – 3, p. 10. (1969).

[4] LUCS, P.: An Accumulator Chip. IEEE Trans. on Computers, Vol. C – 18, pp. 105 ... 114. (1969).

[5] The Staff of the Computation Laboratory: Synthesis of Electronic Computing and Control Circuits. The Annals of the Computation Laboratory of Harvard University, Vol. 27. Harvard University Press, Cambridge, Mass. (1951).

Korte technische berichten

Laser checks air pollution

A laser-light technique devised by two Bell Laboratories scientists promises to provide rapid, precise, 'on-the-spot' identification of pollutant gases in the atmosphere. The system was developed to monitor nitrogen oxides, the major pollutant contents of automobile exhausts and chimney emissions.

The gases absorb some energy from a laser beam at certain frequencies, and careful measurement can determine the nature and volume of pollutants. Experiments in car parks in New Jersey have shown nitric-oxide concentrations ranging from 0.1 ... 10 parts per million.

A specially developed '*spin flip*' Ramen laser provides tunable radiation in the 5-6 μm and 9-14 μm wavelength ranges, which cover the absorption bands of most known pollutants.

ESIP American Newsletter.

Varia

Symposium 'Walsh-functions' 1972

Op 27 ... 29 maart 1972 zal opnieuw een Walsh-function symposium plaatsvinden in Washington D.C. De organisatie is in handen van het Naval Research en de IEEE Electromagnetic Comptability Group.

Bijdragen over de resultaten van onderzoek op het gebied van de toepassing van Walsh-functies en andere in de communicatietechniek voorkomende functies, behandeling van radarsignalen, beeldtechniek, patroonherkenning, seismologie, enz. dienen *vóór 15 december 1971* te worden ingediend.

Nadere gegevens zijn te verkrijgen bij het Secretariaat van ondergetekende, T.H. Delft. Tel.: (01730) 3 32 22, tst. 6193.

Prof. dr. ir. J. L. Bordewijk.

Uit het NERG

Administratie van het NERG: Postbus 39, Leidschendam. Giro 94746 t.n.v. penningmeester NERG, Leidschendam. Secretariaat van de Examencommissie-NERG: Van Geusaustraat 151, Voorburg.

Ledenmutaties

Voorgestelde leden:
Prof. ir. O. W. Memelink, Twickellaan 11, Enschede.

Nieuwe adressen van leden:
Ir. F. P. van Enk, 3 Colstan Crt, Mount Eliza, Vic 3930, Australië.
Ir. G. L. Reijns, Goeverneurkade 5, Voorburg Z.H.
Ir. M. Skaliks, Schumannlaan 17, Enschede.

Overleden:
Ir. P. H. Boukema, Timorstraat 21, Delft.